Functions

A function is a block of code which only runs when it is called.We can pass data, known as parameters, into a function.They are important for reusing code: Define the code once, and use it many times.

Function declaration and definition

A C++ function consist of two parts:

Declaration: It consists of return type, the name of the function, and parameters.

Definition: It is the body of the function (code to be executed).

Ex:

```
 // Function declaration
void myFunction();

// The main method
int main() {
  myFunction();  // call the function
  return 0;
}

// Function definition
void myFunction() {
  cout << "I just got executed!";
}
```

Calling a function

While creating a C++ function, we give a definition of what the function has to do. To use a function, we have to call or invoke that function.

When a program calls a function, program control is transferred to the called function. A called function performs defined task and when it's return statement is executed or when its function-ending closing brace is reached, it returns program control back to the main program.

Ex:

```
// Create a function
void myFunction() {
  cout << "I just got executed!";
}

int main() {
  myFunction(); // call the function
  return 0;
}
```

Function Arguments

If a function is to use arguments, it must declare variables that accept the values of the arguments. These variables are called the **formal parameters** of the function.

The formal parameters behave like other local variables inside the function and are created upon entry into the function and destroyed upon exit.

While calling a function, there are two ways that arguments can be passed to a function -

Type	**Description**
Call by Value	This method copies the actual value of an argument into the formal parameter of the function. In this case, changes made to the parameter inside the function have no effect on the argument.
Call by Reference	This method copies the reference of an argument into the formal parameter. Inside the function, the reference is used to access the actual argument used in the call. This means that changes made to the parameter affect the argument.

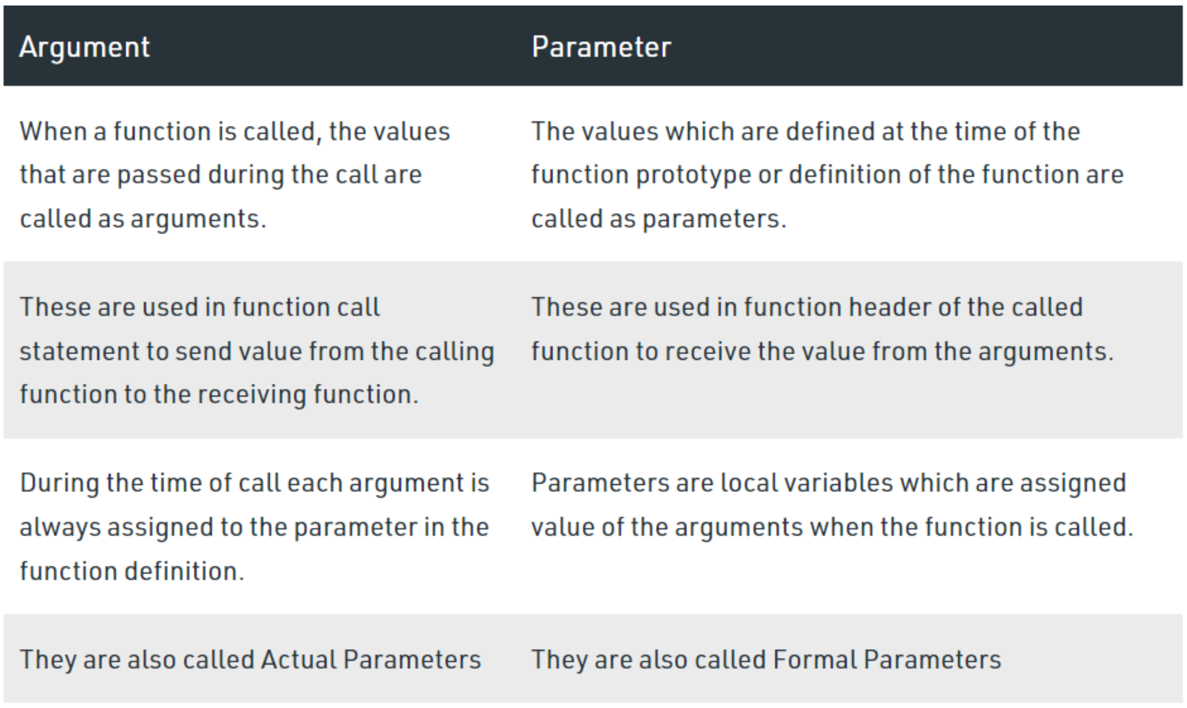
Function parameters

Information can be passed to functions as a parameter. Parameters act as variables inside the function. They are specified after the function name, inside the parentheses. We can add as many parameters as we want, just separate them with a comma.

Syntax:

```
void functionName(parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Difference between Argument and parameter



Argument	Parameter
When a function is called, the values that are passed during the call are called as arguments.	The values which are defined at the time of the function prototype or definition of the function are called as parameters.
These are used in function call statement to send value from the calling function to the receiving function.	These are used in function header of the called function to receive the value from the arguments.
During the time of call each argument is always assigned to the parameter in the function definition.	Parameters are local variables which are assigned value of the arguments when the function is called.
They are also called Actual Parameters	They are also called Formal Parameters